www.icgst.com

Automatic Conversion of Natural Language into Relational Database Queries

Prof. Dr. Kamal ElDahshan,
Faculty of sciences, Department of Math Al-Azhar University,
Khalid Walid, Mennatullah Walid, Mohamed Ali
Faculty of Business Information Technology Canadian international college Cairo Egypt
dahshan@gmail.com, Khaled_walid@cic-cairo.com, Mennatullah_walid@cic-cairo.com, m_ali_ali@cic-cairo.com

Abstract

This paper presents possible solutions for a problem faced by casual users in technology fields. This problem is the extraction of information as experienced users through writing queries. The main idea of the project is to convert the user's natural language to SQL (structured query language) queries. This is done by applying an algorithm to extract the required data from user's input and then applying the query to the database to extract the needed information. This should be most effective in case of ad-hoc queries. The main goal of the project is to enable the non-specialized user to extract whatever information he needs from the database. As long as the user requires privileges by adding a layer that improves the HCI (human computer interaction) through the use of NLP (natural language processing). This technique eases the extraction of information from the database and lessens the stress and pressure on specialized users, thus increasing the overall efficiency.

Keywords: *SQL; NLP; Dependency parsing; Parts of speech tagging; Natural language bank; Data dictionary; Natural language query; NLIDB.*

Nomenclature:

HCI	human-computer interaction
SQL	structure query language
NLP	natural language processing
IT	information technology
DBMS	database management system
NLIDBs	natural language interface databases).
NLI	natural language interface
PK	primary key
FK	foreign key
NLQ	Natural Language Query
CS	computer science
WH	type of questions tools that starts with WH such as "what, when"

1. Introduction

One of the problems facing current programmers, database developers, people working in IT generally, or even students is the complicated syntax. Although some say that the more experienced a developer is, the faster he can grasp that syntax, however not everyone is a developer who knows how to code applications and databases, as a result, the problem of the syntax still applies to most software packages such as: MySQL DBMS, Oracle DBMS, and programming languages such as: C#, C++, etch. Despises the currently used computer based information retrieval techniques to help organizations and business being effective even when dealing with large amounts of data especially in relational databases the user still needs to understand and master the logic, rules and syntax of database in order to use it. The proposed solution use some functions of natural language processing in order to help the casual users to interact effectively with the database by letting them type their request for information in English natural language then process that input using various methods in order to construct a query that can retrieve the requested information. Basically, the goal is to improve the HCI (human-computer interaction). HCI is the study of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings. As its name implies, HCI consists of three parts: the user, the computer itself, and the ways they interact together, but in this case, the main concern is improving the interactions between users and databases by allowing casual users to freely interact with the database management systems in order to retrieve any information they need through a natural language interface.

The rest of the paper is organized as follows; section 2 introduces the literature review. Section 3 presents the proposed solution. Section 4 is the methodology. Section 5 explains the implementation. Section 6 is the pseudo code. Section 7 presents the result analysis. Section 8 is the conclusion and future work.

2. Literature review

Since the end of the 1960s there has been a wide number of research papers indicating the theories and execution of NLIDBs (natural language interface databases) and other platforms that try to use NLI (natural language interface) such as Search Engines and similar software in order to implement natural language processing interfaces for information retrieval. 'Rendezvous' system appeared in late seventies, in which end users could access databases through relatively unrestricted natural language. In this system, the main consequence is placed on query description and fetching user's conversation when there is difficulty in parsing user query. [5], 'Querix' is a domain-independent natural language interface that uses clarification dialogs to query ontologies. The approach is simple and does not use any complex semantics-based technologies. Compared to a full natural language processing (NLP) engine, it does not try to resolve natural language ambiguities, but asks the user for clarification [4], 'Panto' is a Portable Natural Language Interface to Ontologies, it utilizes an off-the-shelf statistical parser Stanford Parser to deal with the first major obstacle. Multiple existing techniques and tools are integrated to interpret parse trees of natural language queries into 'SPARQL'. [9]. 'Step' provides a natural language interface for dealing with relational database in a purely textual interaction [3], 'Quest IO' System works by converting natural language input into a formal semantic query. [8], 'Editie' is a multi-lingual (Portuguese, French, English, Spanish) natural language front-end for relational databases, it answers written questions about tourism resources by transforming them into SQL queries. The answer depends on the type of question. [7], 'Team' was developed in 1987 for portability issues. TEAM was designed to be easy and configurable by database administrators with no knowledge of NLIDBs. [5].

3. The Proposed Framework

Many of the previous systems or frameworks that wanted to apply natural language processing to user's input to transform it to SQL query failed to do so or didn't achieve required performance, one common factor between them is their constant attempt to move away from the human factor in the NLP process. The proposed framework doesn't try to tackle the full or actual problem of natural language processing instead it aims to reduce the communication gap between the user and the database as shown in figure 1 below by trying to form a relation between user's input and the information he needs from the database by following specific algorithms. The main objective of this framework is to allow the casual user to interact without many limitations (Syntax, experience constraints) with the database. The main point of the application is to allow the human factor to monitor the natural language processing part in order to form the necessary core data for the layer to function, this is done by assigning a database administrator role. The database administrator will have the right to manage what is called a natural language bank and a data dictionary, the said elements are crucial for the success of the layer. The natural language bank will be used as the source that is constantly referred to in the process of parsing the user input and transforming it to query. Once the layer (proposed framework) connects to the database it automatically constructs a list of certain elements in the database as an example the table names and the existing attributes within each table. The task of the database administrator is to write the description of each item in the data dictionary [1] as an example (std_name = student name) also to determine the relations concerned with this table and a preparation of the mapping process, in addition to performing the mapping process which will link the items in the natural language bank with their counterparts in the data dictionary. The main components of the proposed framework are as follows: Input Parser, Natural language Bank, Mapper, Data Dictionary and Query Builder. This layer is designed to function on any relational database built using MySQL DBMS (Database management system).

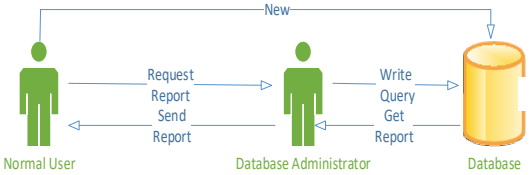

Figure 1: Proposed Framework

4. Methodology

The main points needed to achieve success in the layer construction are as follows: in order to retrieve the information successfully from the database a query needs to be constructed to get the information, which means that the user input must be separated into the elements forming the SQL Query, by specifying which elements will be in the select clause, the from clause and the where clause of the SQL query. Deciding which elements will be in the from clause will happen using the structure of the application database and a few pre-set algorithms. The structure of the application is illustrated in figure 2. First, the user input is processed. Once the main items in the user input are specified using algorithms to decide the type of the input whether it was a declarative phrase or imperative it is compared to the natural language bank contents then mapped to the data dictionary contents. The result of the mapping process is: determining the required information that the user asked for and where it can be found (in which tables they exist). These items are then sent to the query builder component in order to build the query needed for extracting the information.

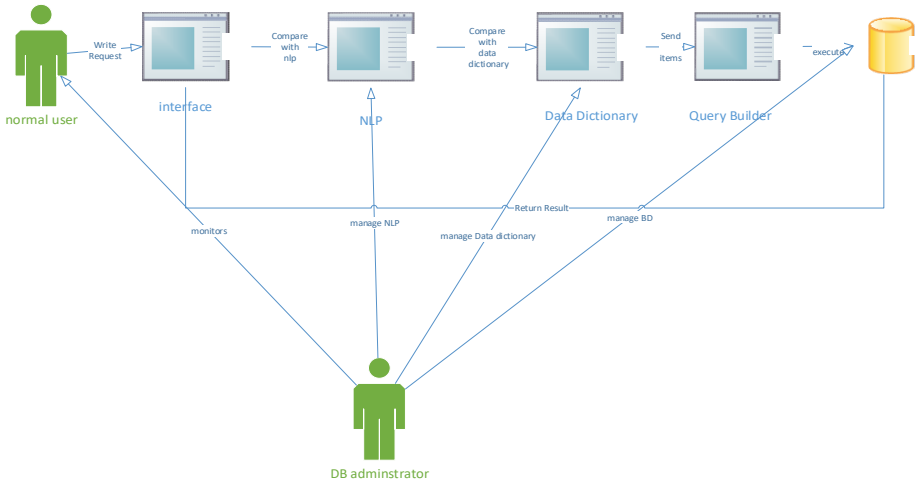

Figure 2: General Structure

5. Implementation

The most important factor in the implementation is how to specify the elements of the query namely the select clause and, from clause and the where clause. First the process to find the elements of the where clause will be explained in the following example (in the database of the layer there will be a table containing all the databases information that the layer will connect to. Each record of said table will be connected to another table containing information about all tables that exist in a database. Each record of this table will be connected to entries in another table called 'data dictionary' which will have the name, the attributes of the table, their description, data type and role in the table which specifies if the element is an attribute or a PK (primary key) or FK (foreign key). Once the user enters his input and it is sent to the input parser for processing, a mapping process will occur after finding which elements the user needs, the mapping process will happen between the natural language bank and the data dictionary. Using that, we will be able to determine the original table that contains the needed element, thus, forming the relation between the tables. The users input will be received by the interface then sent to the input parser where the following will occur: the tokenization of the input (user input will be split into words each given a number in order to better classify and deal with the input) [12]. The tokenized input will be transformed into lower case characters in order to form a standard for processing the input, after that starts the process of filtering the input by comparing it against what we call a stop list which contains elements that were determined to be of low value to the input parser and will only serve to increase the parsing time and make it more confusing/harder to execute, any elements that exist in the stop list will be removed from the input. The query is generated by identifying the lexical relations of the elements of the NLQ (Natural Language Query) [6]. Following that comes the separation and the specification of the phrase elements to (prepositions, question words, comparison operators, articles, determiners, verbs, etc.….) which is called lexical analysis [11] using a custom-made library that contains the said elements. In the parsing process the input of the user will be divided into the said groups, each of these groups hold a meaning, some of them will be used as a mark or a trigger for the condition, others will be used as the value of the condition itself or simply as the needed information, as an example a college database will have the following tables: students, majors, grades; the user question will be : "Display all the student belonging to CS major and have GPA higher than 3.5", this will be parsed as follows: "student" will be classified as a key item, "belonging" will be classified as a determining clause, "CS" will be classified as a value, "major" will be classified as a condition word, "and" will be used to refer to an extended condition. Another example would be: "Display employee's name and salary where/whose salary is between/within 3000 to 5000", same classification rules will apply, this can be achieved by finding a pattern with user's question in order to analyse it and find out when does the user type the condition for his request and using it to determine the condition or by following grammatical rules such as the ones proposed in dependency parsing. As for analysing user question patterns, database tables were distributed amongst technical and non-technical people and they were asked to request any information they might need from those sets of tables in English form. After analysing their information requests it was found that seventy percent of test subjects used or depended on what's generally known as WH question words(how, what, where etc…), in order to ask for information as an example: "View the student names who got more than 40 in their course work" others used what's called as yes/no questions as an example: "Did the student x fail their CS course" and the last group of test subjects used order form to request

the information they wanted as an example: “Show students that are currently registered in CS course”. Generally speaking, these are the most common ways of requesting information so a set of algorithms were made for each one of them in order to separate the needed information from the condition and its value which are used to display the needed information. Dependency parsing is about parsing the context of the grammar which determines the structure of the phrase by establishing the relations between a word and its dependencies, such grammar is called dependency grammar [2]. The results of the input parser are words in their respective groups that will be sent to the natural language bank for comparison, as an example: the nouns or the words will be sent to the natural language bank to check if they exist and what are their relations and positions, if the said items don’t exist, the database administrator can edit the natural language bank to add them, if they exist then the results of the comparison are sent to another component which is the mapper. The mapper is responsible for the mapping process which first occurs after the complete construction of both the data dictionary and the natural language bank and it is done by the database administrator, in case of any additions to the natural language bank or to the data dictionary, modifications must be done to the mapping process in order to match their updated content, as a precaution a comparison is always made between results of natural language bank and the data dictionary items whenever a user asks for information. The data dictionary components receive the results of comparing the parsed input with the natural language bank content as a preparation for the final comparison. The final comparison will be between the received results and the items existing in the data dictionary. The result of the final comparison between natural language bank and data dictionary will be the required data items to use in forming the query to extract the required information from the database, this process will happen in the query builder component. The sequence of the steps that the user will take to get at the needed information are illustrated in figure 3.

6. Pseudocode

The core of the implementation is made using the following pseudocode.

Receive input string from user
Decompose the string into separate words
Compare each single word with the natural language bank
For each word in string
If word is in natural language bank then
    Save them in a different variable
    Check their classification in bank
    Return their classification
Else if words in stop list then
    Remove them from input string
End if
End for
Receive the processed input based on classification
Then processed input gets sent to sentence builder
After removing complex parts of input the sentence gets transferred into its simplest structure
After the sentence Is in simple structure its re analysed and classified once more to determine which part goes into select and what are the parts assigned to where main conditions
For each word in string
If word is mapped with data dictionary then
    Retrieve field name and description
Else
    Return error message
End if
Swap the existing words with their data dictionary counter parts
Then apply a query to database to retrieve all relations in a database
Using the new input (which has been swapped with data dictionary) and compare it with table fields to know the tables these fields belong to for the from part in query
Now use the relation gotten earlier from applying the query to database and loop over it to find the relation of the tables with each other to add it in the where condition
And thus, the query is formulated and sent to database to be executed.
Below is figure 4 which is the application interface that is used to get the results and figure 5 which is one of the target database application is tested on

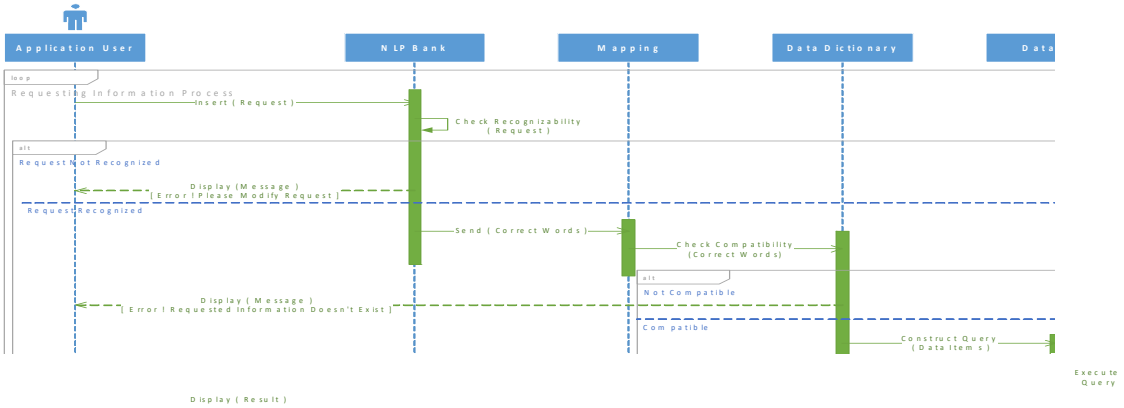

Figure 3 User Request Sequence

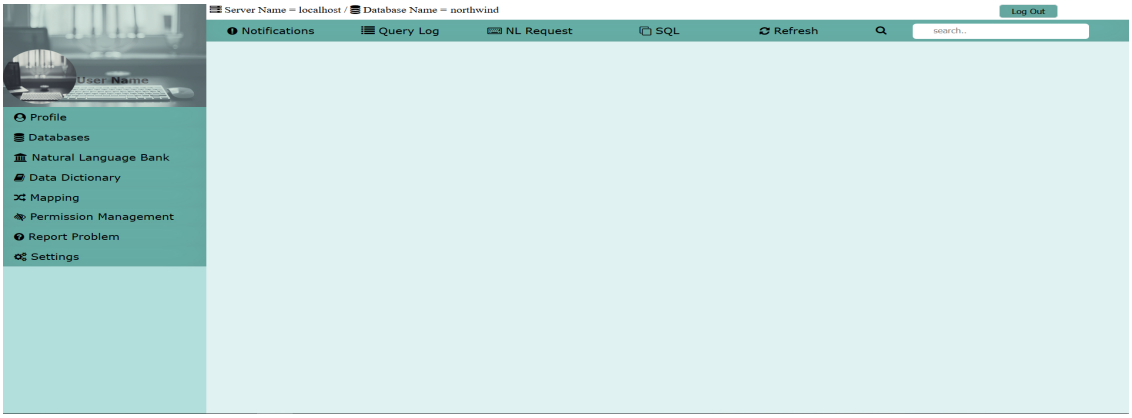

Figure 4 Application Interface

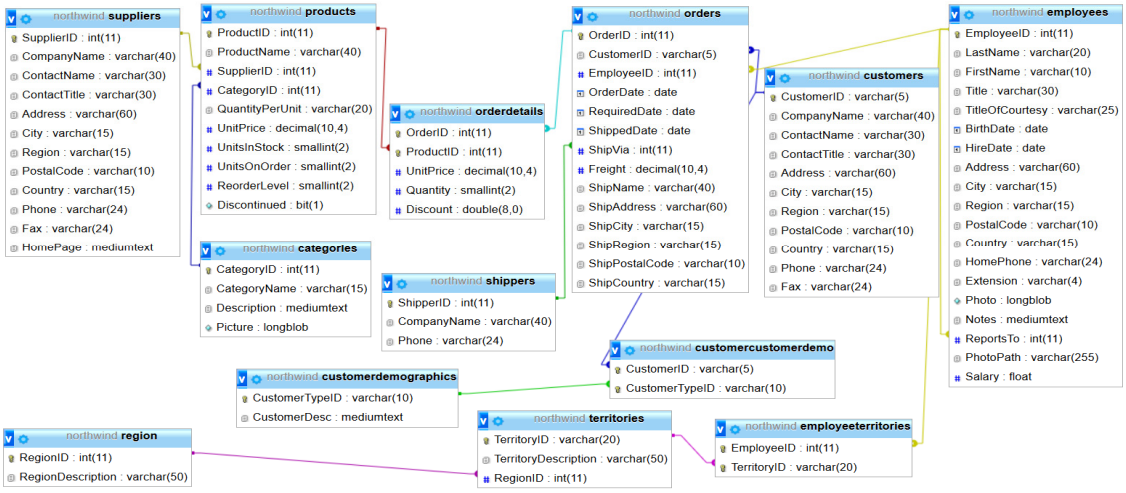

Figure 5 North wind Database

7. Result Analysis

In order to calculate the effectiveness of this layer it was tested on three different databases consisting of 10 tables each and sample data was collected from people working in the IT industry (information technology) and casual users by asking them to ask for any information they need from the said databases. All the results are based on actual data that was fed into the main components of the layer. A natural language bank and a data dictionary are made for each of the said databases for testing purposes. A total of fifty queries were tried for each database sample and the results were

classified as follows. Simple queries are ones that don't require conditions their success rate is ninety-five percent. Higher level queries between tables that are directly related had a success rate of eighty percent. Higher level queries between tables that aren't directly related had a success rate that varies from sixty to eighty percent based on the information that has been previously fed to the layer by the database administrator. The cause of low success rate is the complexity of separating the user input into different classifications, as an example: the parts of input that are related to the select clause, from clause and most importantly the where clause. Another cause for low success rate is the fact that indirect relations between tables are hard to solve dynamically using algorithms without the help of the database administrator also the execution time of the queries isn't considered as fast as it could be due to the need for mapping the user input each time, but it can be improved by using better algorithms for the comparison and the parsing process, also the overall efficiency of the layer can be improved based on the data that is fed into it by the database administrator and accuracy of the queries can be further improved assuming the improvement of the algorithm used to separate the input parts that belong to the select clause and those that belong to the where clause. Another way of improving the results is offering examples for the casual users to follow, doing so will increase the results' accuracy by ten percent for each category it will also speed up the execution time, more over by feeding more accurate and specific data into the layer such as increasing the stop list content and the content for the specific library that is used for parts of speech tagging as it has the crucial task of assigning part of speech tags to the user input [10] in order to further analyze it to get the needed information which would lead to another increase in the overall accuracy of the layer.

.

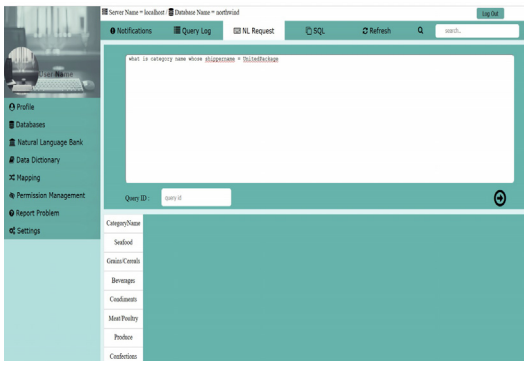

Figure 6 Query Results

Below are screenshots of results gained by the application in figure 6,7.

In figure 6 the to get the query result the application made had to bring information from 5 different tables while connecting them using the algorithm and in figure 7 its just information from a single table.

8. Conclusion and future work

Finally, this paper shows how the layer can translate the English statement (natural language) to the queries in an efficient way. The required knowledge and training period is decreased for the employees, so the overall usability is increased because any type of user (technical or non-technical) can use this layer and get the needed result or information, and the work pressure on the IT expert or database administrator is reduced. The important aspect of this layer is the functionality of updating the data dictionary for the synonyms of words, and the natural language bank to improve the program, and reducing the errors during its use. The layer provides user input and theme options for the user to make it as easy and user-friendly as possible. As an extension to the improvement of the HCI in this topic it is planned to test two more methods for processing the user input and improve the accuracy of the results in order to improve the overall efficiency of the proposed layer. As a future work after achieving an acceptable percentage of accuracy an implementation of the speech recognition feature for the layer will be made in order to improve its HCI, as well as implementing new algorithms to improve the performance of the input parser, especially parts of speech tagging component and also to improve the mapping component which would lead to a dramatic increase in the layer's performance. Finally improving the technique used to automatically investigate the relations between tables would lead to the layer being able to handle queries with higher complexity levels.

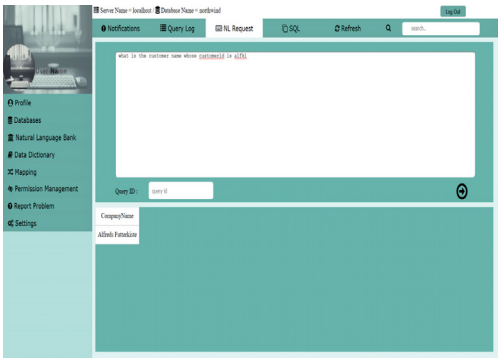

Figure 7. Simple Query result

9. References

[1] A. M. O. a. K. C. Alghamdi, "Natural Language Interface to Relational Database (NLI-RDB)," Advances in Computational Intelligence Systems. Springer International Publishing, pp. 449-464, 2017.

[2] P. M. D. R. Radev, "Algorithms for Information Retrieval and Natural Language Processing tasks," 2008.

[3] M. Minock, "A phrasal approach to natural language interfaces over databases," in International Conference on Application of Natural Language to Information Systems, Springer, 2005, pp. 333-336.

[4] E. a. B. A. a. Z. R. Kaufmann, "Querix: A natural language interface to query ontologies based on clarification dialogs," 2006.

[5] Deepshikha, ""A Survey of Natural Language Query Builder Interface to Database"," International Journal (International Journal)., 2015.

[6] S. R. Gupta A, A Novel Approach to Aggregation Processing in Natural Language Interfaces to Databases., Hyderabad, India.: Language Technologies Research Centre International Institute of Information Technology, , 2012.

[7] P. a. 1. Reis, "Edite-A Natural Language Interface to Databases A new dimension for an old approach.," in Information and Communication Technologies in Tourism, Vienna, 1997.

[8] C. a. X. M. a. Z. Q. a. Y. Y. Wang, "A portable natural language interface to ontologies," in European Semantic Web Conference, 2007.

[9] V. a. D. D. a. B. K. Tablan, "A natural language query interface to structured information," in European Semantic Web Conference, 2008.

[10] B. a. R. S. V. Sujatha, "Natural Language Query Parser using First Order Logic for Querying Relational Databases," International Journal of Computer Applications, pp. 43--48, 2016.

[11] S. Prabhdeep Kaur, "CONVERSION OF NATURAL LANGUAGE QUERY TO SQL," International Journal of Engineering Sciences & Emerging Technologies, vol. 8, no. 4, pp. 208-212, 2016.

[12] S. D. S. S. Prasun Kanti Ghosh, "Automatic SQL Query Formation from Natural Language," in International Conference on Microelectronics, Circuits and Systems, 2014.

Biography:

Prof. Kamal Abdelraouf ElDahshan He is a professor of Computer Science and Information Systems at Al-Azhar University in Cairo, Egypt. An Egyptian national and graduate of Cairo University, he obtained his doctoral degree from the Universitè de Technologie de Compiègne in France, where he also taught for several years. During his extended stay in France, he also worked at the prestigious Institute National de Tèlècommunications in Paris. Professor ElDahshan has extensive international research, teaching, and consulting experiences have spanned four continents and include academic institutions as well as government and private organizations. He taught at Virginia Tech as a visiting professor; he was a Consultant to the Egyptian Cabinet Information and Decision Support Centre (IDSC); and he was a senior advisor to the Ministry of Education and Deputy Director of the National Technology Development Centre. Professor ElDahshan is a professional Fellow on Open Educational Resources as recognized by the United States Department of State and an Expert at ALECSO as recognized by the League of Arab States.

Khalid Tokatly is a graduate of the Faculty of Business Information Technology Canadian international college Cairo Egypt. He is interested in NLP, AI, Business Intelligence and Database Systems.

Mennatullah Tokatly is a graduate of the Faculty of Business Information Technology Canadian international college Cairo Egypt. She is interested in NLP, AI, Business Intelligence and Database Systems.

Mohamed Farrag is a graduate of the Faculty of Business Information Technology Canadian international college Cairo Egypt. He is interested in NLP, AI, Business Intelligence and Database Systems.